REMARKS

Claims 10 and 23-25 have been cancelled to focus prosecution on claimed embodiments concerning network interfaces having updatable filter address tables.

Claims 4, 13, 15, 16,18, and 19 have been amended to correct clerical errors or to claim a NIC having an updatable node address memory or updatable filter table.

Claims 1-9, 11-22, and 26-27 are presently active.

35. U.S.C. §112

The Action noted a missing "base" reference in claim 4; the claim has been amended accordingly. The Action noted a grammatical error in claim 18; the claim has been amended accordingly.

The Action also refers to the claims as generally being a narrative and indefinite, but does not specifically identify a problem in the claims. Applicant is not sure what is meant by this characterization, but if this rejection does not relate to, or is not addressed by the claim amendments and/or cancellations, the Office is respectfully requested to contact Steven D. Yates at 503-264-6589 to discuss this issue.

35. U.S.C. §103(a)

Claims 1-3, 6, 11-14, 20-22 and 26-27 stand rejected under §103(a) as being obvious over Amdahl U.S. Patent No. 6,253,334) in view of McIntyre (U.S. Patent No. 6,381,218). Applicant traverses these rejections as claimed embodiments recite limitations not taught or suggested by the cited documents.

For example, **claim 1** recites an API having an "update node address function" that updates the node address stored in a base driver and a receive address filtering

table for a network interface (NIC), e.g., storage of the Media Access Control (MAC) address allowing a NIC to recognize data intended to be received by it (see, e.g., Specification at page 5 lines 13-16). In one embodiment, by way of the claimed API, if there is a failure of an active NIC, a redundant NIC can cover for the failed NIC by having the redundant NIC rewrite its MAC address in its filter table to include the MAC address of the failed NIC (the redundant NIC will see incoming traffic as being intended for it) (see, e.g., Specification at page 9 lines 18-25 and page 11 lines 1-6).

Applicant submits Amdahl does not teach the recited updating NIC filter tables. Instead Amdahl teaches using a switch 1420 (FIG. 13) that is aware of MAC addresses of Amdahl NICs, where the switch "routes incoming packets to whichever of NICs 1394-1396 has a MAC address matching the destination address of the packet" (col. 21 lines 58-60). If there is a NIC failure, Amdahl teaches "terminating communication with the unreachable/failed NIC" and load-balancing further traffic with remaining NICs (col. 23 lines 4-10). Amdahl teaches that the load balancing is accomplished at the Open System Interconnection (OSI) "data layer" by altering the MAC address ***in a packet header*** to conform to the NIC intended by Amdahl to receive the subsequent traffic (col. 22 lines 25-32).

That is, Amdahl teaches rewriting incoming packets to direct them to a redundant NIC instead of the recited alteration of a redundant NIC's filtering table address.

This is not what is claimed.

Regarding the Office's citation to Amdahl at col. 8 lines 54-62 as teaching the recited "update node address function," this is not correct. The cited portion of Amdahl concerns changing multicast address lists when a NIC fails. In Amdahl, when a NIC

fails, a multicast update call is performed to identify the covering ("switched-over" col. 8 line 61) NIC. Updating multicast address lists has no relation to the recited "update node address function."

Regarding combination with the cited portion of McIntyre, even if we assume as suggested by the Office that McIntyre teaches the recited stored node address, since Amdahl teaches rewriting packets on fail-over, rather than updating the NIC filter table with a new MAC address as recited in claimed embodiments, the suggested combination of references can not teach or suggest the recited embodiments.

Regarding the rejection of **claim 11**, as discussed above for claim 1, combination of Amdahl and McIntyre fails to teach or suggest the recited API for updating a receive address filtering table for a NIC. Consequently, the cited documents can not teach or suggest claim 11's update node address API function which may direct a second, e.g., covering, NIC to store the node address, e.g., MAC address, for a failed NIC.

Regarding the rejection of **claim 6**, it also recites updating a receive address filtering table as does claim 1. Applicant submits claim 6 is allowable for at least the reasons discussed above for claim 1.

Claim 13 was rejected per the rejections of claims 1 and 11. Notwithstanding the rejection, claim 13 has been amended to recite the API providing for updating a receive address filtering table network interface as discussed above for claim 1. Applicant submits claim 13 is allowable for at least the reasons discussed above for claim 1.

Regarding the rejection of **claim 15**, to more closely relate claim 15 with the subject matter of claim 1, limitations from claim 16 were moved into claim 15. Corresponding amendments were made to **claim 19**. Applicant submits these

amendments moot the rejections and claims 15 and 19 are allowable for at least the reasons discussed above for claim 1.

Regarding the rejection of **claim 20**, as discussed above for claim 1 with respect to Amdahl's figure 13, Amdahl does ***not*** change the MAC address of NIC hardware. Instead, Amdahl teaches rewriting incoming packet headers (col. 21 lines 58-60). Consequently the cited documents can not be combined as suggested and therefore, as with claim 1, they cannot render the claimed embodiments obvious.

Regarding the rejection of **claim 23**, as discussed above, Amdahl fails to teach or suggest the recited

Claim 26 stands rejected per claim 1 and Applicant therefore submits it is allowable for at least the reasons as discussed above for claim 1.

Regarding the rejections of dependent claims 2-5, 7-9, 12, 14, 15, 16-18, 21, 22, 24, 25, and 27, their rejections are not being specifically addressed at this time in order to focus prosecution on their base claims. Applicant submits these dependent claims are allowable for at least the reason as depending from an allowable base claim.

CONCLUSION

Based on the foregoing, it is submitted that that all active claims are presently in condition for allowance, and their passage to issuance is respectfully solicited.

The Examiner is requested to contact the undersigned by telephone if it is believed that such contact would further the examination of the present application.

Respectfully submitted,

Date: May 9, 2003

Steven D. Yates
Patent Attorney
Intel Corporation
Registration No. 42,242
(503) 264-6589

c/o Blakely, Sokoloff, Taylor & Zafman, LLP
12400 Wilshire Boulevard
Seventh Floor
Los Angeles, CA 90025-1026

APPENDIX A

Claims As Amended

1. (Unchanged) An application programming interface (API) for enhancing data network communication, comprising:

an identify address function including programming instructions for identifying a stored node address stored by a base driver for a network interface associated with the base driver; and

an update node address function including programming instructions for directing the base driver to update the stored node address with a new node address in a configuration storage of the base driver, and in a receive address filtering table for the network interface.

2. (Unchanged) The API of claim 1, wherein the identify address function includes submitting a request to the base driver, to which is received a response including the node address stored by the base driver.

3. (Unchanged) The API of claim 1, wherein the identify address function includes programming instructions for inspecting the configuration storage of the base driver, such storage having an entry identifying the stored node address.

4. (Once Amended) An API according to claim 1, further comprising:

a driver identification function including programming instructions for sending an identity-check request to the base driver, said base driver providing a response selected from a group consisting of: a predetermined identifier, a base driver revision number, and an identification of a vendor of the base driver.

5. (Unchanged) An API according to claim 4, wherein the predetermined identifier is a copyright string for the vendor of the base driver.

6. (Unchanged) An article of manufacture, comprising a computer readable medium having encoded thereon programming instructions capable of directing a processor to perform operations of:

an identify address function for identifying a stored node address stored by a base driver for a network interface associated with the base driver; and

an update node address function for directing the base driver to update the stored node address with a new node address in a configuration storage of the base driver, and in a receive address filtering table for the network interface.

7. (Unchanged) An API according to claim 1, further comprising:

a first transmission function including programming instructions for re-transmitting data, received in a compatible format from a network source, in an incompatible format to a network destination; and

a second transmission function including programming instructions for re-transmitting data, received in the incompatible format from the network destination, in the compatible format to the network source.

8. (Unchanged) An API according to claim 7, further comprising:

a report capabilities function including programming instructions for sending the base driver a request to have the base driver report its capabilities;

a receive capabilities function including programming instructions for receiving a response including said capabilities;

wherein the incompatible format is formatted according to said capabilities.

9. (Unchanged) An API according to claim 7, further comprising:

a virtual LAN function including programming instructions to direct the base driver to enter a desired virtual LAN operative state; and

a disconnect function including programming instructions to notify the base driver that the API has concluded communications with the base driver.

10. (Cancelled)

11. (Unchanged) An API according to claim 1 for providing transparent fail-over from a first network interface to a second network interface, further comprising:

a status function including programming instructions for polling a first base driver for the first network interface to detect a failure of said first network interface;

wherein the update node address function includes a function to direct a second base driver for the second network interface to store the node address of the first network interface as the stored node address for the second base driver.

12. (Unchanged) An API according to claim 11, in which a Novell ODI compliant network is utilized for network communication, and wherein the update node address function uses at least one ODI MLID Control Routine.

13. (Amended) An article of manufacture, comprising a computer readable medium having encoded thereon instructions to direct a processor to perform an API having:

an identify address function for identifying a stored node address stored by a base driver for a network interface associated with the base driver;

an update node address function for directing the base driver to update the stored node address with a new node address;

a status function in communication with a first base driver for the first network interface to detect a failure of the first network interface; and

a failover function to direct a second base driver for the second network interface to store the node address of the first network interface as the stored node address for the second base driver, and to store the node address of the first network interface in a receive address filtering table for the second network interface.

14. (Unchanged) An API according to claim 1 for providing transparent load balancing of data transmissions directed towards the network interface by distributing such data across a second network interface, further comprising:

a queue monitoring function including programming instructions for detecting a workload of the first network interface; and

a distribution function including programming instructions for routing a portion of said data transmissions through the second network interface, said distribution function utilizing the update node address function to associate the node identifier of the first network interface with the second network interface.

15. (Once Amended) A networking method, comprising:

receiving first network traffic with a protocol stack;

sending said first traffic to an intermediary layer;

routing said first traffic to a virtual interface driver;

repackaging said first traffic by the virtual interface driver, and providing said repackaged traffic to a virtual protocol stack;

sending said repackaged traffic to the intermediary layer;

routing said repackaged traffic by the intermediary layer to an interface driver for a network interface having a node address memory;

identifying a failed network interface having a node address; and

storing the node address in the node address memory.

16. (Once Amended) A method according to claim 15, further comprising:

routing network traffic for the failed network interface through the fail over network interface.

17. (Unchanged) A method according to claim 16, further comprising:

wherein said first network traffic is received in a first protocol format, and said repackaged traffic is in a second network protocol format different from the first protocol format.

18. (Once Amended) A method according to claim 16, wherein locating the fail over network interface comprises:

submitting a node identification request to a base driver for a potential fail over network interface; and

receiving a response from said driver, said response including an authentication string;

verifying said authentication string has a predetermined value before said potential fail over network interface is used as the fail over network interface.

19. (Once Amended) An article of manufacture, comprising a computer readable medium having encoded thereon instructions to direct a processor to perform the operations of:

receiving first network traffic with a protocol stack;

sending said first traffic to an intermediary layer;

routing said first traffic to a virtual interface driver;

repackaging said first traffic by the virtual interface driver, and providing said repackaged traffic to a virtual protocol stack;

sending said repackaged traffic to the intermediary layer;

routing said repackaged traffic by the intermediary layer to an interface driver for a network interface having a node address memory;

identifying a failed network interface having a node address; and

storing the node address in the node address memory.

20. (Unchanged) A method for redundant networking in a network environment, comprising:

determining operative status of a first network interface having a first driver, and of a second network interface having a second driver with a driver memory for storing a MAC address for said second interface;

if the first network interface is inoperative, instructing the second driver to store the first network interface MAC address in the driver memory to allow processing by the second network interface of network traffic bound for the first network interface;

directing the second driver to activate the second network interface; and

directing the first driver to deactivate the first network interface.

21. (Unchanged) A method according to claim 20, in which the network environment is a Novell based network, and wherein ODI commands are issued to said first and second drivers.

22. (Unchanged) A method according to claim 21, further comprising:

receiving first network traffic by a protocol stack;

forwarding said first network traffic to a LSL;

routing said first network traffic from the LSL to a virtual MLID, and deriving second network traffic from said first network traffic;

providing said second network traffic to a virtual protocol stack; and

forwarding said second network traffic to the LSL.

23. (Cancelled)

24. (Cancelled)

25. (Cancelled)

26. (Unchanged) A system, comprising:

means for identifying a stored node address stored by a base driver for a network interface associated with the base driver; and

means for directing the base driver to update the stored node address with a new node address.

27. (Unchanged) A system according to claim 26, further comprising:

means for re-transmitting data, received in a first format from a network source, in a second format to a network destination; and

means for re-transmitting data, received in the second format from the network destination, in the first format to the network source.